Discriminant Features Extraction by Predictive Neural Networks

B. GAS, J.L. ZARADER, C. CHAVY, M. CHETOUANI
Laboratoire des Instruments et Systèmes d’Île de France (LISIF)
Université Paris VI
BP 164, Tour 22-12, 2ème étage,
4 Place Jussieu, 75252 Paris Cedex 05
FRANCE

Abstract: - In this article we propose an extension of the Neural Predictive Coder (NPC) called NPC-2 applied to speech phonemes recognition; This model is an extension to the non-linear area of adaptive coding systems used in speech processing. We show that it is possible with NPC-2 to take into account class membership informations of the phonemes from the stage of coding. In order to evaluate the NPC-2 coder, a study of Darpa-Timit phonemes recognition is done. Presented simulations put in obviousness an improvement of the classification, relatively to currently used coding methods.

Key-Words: - Phoneme Recognition, Neural Networks, Neural Predictive Coding.

1 Introduction
Since few years, speech coding knows a renewal of interest within the scientific community. Actually, at the present time applications in speech recognition give good results but in limited environments (mono-speaker mode, noisy environment, non spontaneous speech, etc.). As a consequence, there is a need to look over the speech signal coding.

The coding stage takes a fundamental place in speech processing since it deals with the discriminative features extraction used by the next stage of recognition (phonetic categorizations for example).

Currently, one finds two main families of coding. The temporal coding methods (LPC, LPCC, LAR, etc.) and the frequential coding methods (FFT filter banks, Cepstre, MFCC, etc.). The cepstral representation using the Mel scale (MFCC coding) is the most often used method because of its robustness.

In our purpose, we focus on short-term non-linearities of the signal and high level knowledge extraction. The neural Predictive Coding (NPC) model proposed by B. Gas and J.L. Zarader in 1996 [1,2,3] allows discriminant features extraction as usual coding methods do. But The NPC-2 extension presented in this article optimizes the coding by incorporating class informations to improve the next pattern recognition stage.

Authors have already tried to improve the coding quality [5,6,7], but usually with a frequential approach, or a cepstral one. The originality of the NPC model is to combine three essential qualities in one predictive model: First, the arbitrary limited number of coding coefficients. As underlined by [12], this problem occurs when using neural networks. Second, the short-term non-linearities modelization and third, the higher level knowledges based feature extraction.

In the first and the second parts, we will describe the NPC and NPC-2 coding models. We will then present in the next two parts the conditions of experiments, the results and the comparisons obtained from other coding methods. The last part will be devoted to the definition of a discrimination ratio measure showing the discriminant features extraction capacity of the NPC-2 coding.

2 The NPC model
The Neural Predictive Coding model is an extension of the LPC traditional coding (Linear Predictive Coding) to the modelling of non linear signals. So it belongs to the temporal coding methods. It is based on a two layers perceptron used as a transversal filter (fig. 2) which is composed of one hidden layer followed by an output layer with a single output cell called the prediction cell. The training stage consists in learning a signal sample (network output) from the previous ones (network inputs).

Let \( y_k = [y_{k-1}, y_{k-2}, \ldots, y_{k-L}]^T \) be a samples sequence (\( L \) is the sequence length called the prediction window width which is equal to the network inputs number). The predictor performs the prediction \( \hat{y}_k \) of the sample \( y_k \):

\[ \hat{y}_k = F(y_k) \]

\( F \) can be viewed as the composition of two functions \( G_w \) (corresponding to the hidden layer) and \( H_w \) (corresponding to the output layer):
\[ F = H_a \circ G_w \text{ with } \hat{y}_k = H_a(z_k) \text{ and } z_k = G_w(y_k) \]

\( w \) denotes the hidden layer weights vector and \( a \) the output layer weights vector. All network weights are usually computed by minimizing a prediction cost function as the quadratic error criterion:

\[
L = \sum_{k} \left( y_k - \hat{y}_k \right)^2
\]

For instance, over all the samples composing a phoneme, one can obtain after learning a function \( F=H_a \circ G_w \) which is a non linear auto regressive model (NLAR) of the phoneme. One can considers \( w \) and \( a \) representative of a set of the phoneme features that could be used for further classification.

In fact, the main drawback of such an approach is the high number of parameters which grows very quickly with the prediction window size and the hidden layer cells number. For example, a 20 x 8 x 1 network structure gives 321 weights (to be compared with the 32 coding coefficients commonly used).

NPC coding allows an arbitrary number of coding coefficients. The key idea is that only the output layer weights are the coding coefficients while the first layer acts as a multi-dimensional filter. This is achieved by creating a hidden layer for each phonemes, the first layer remaining the same for all phonemes (Fig. 1).

Considering now a set of phonemes \( i = 1, \ldots, N \), the cost function previously defined becomes:

\[
L = \sum_{i} \sum_{k} \left( y_{ik} - \hat{y}_{ik} \right)^2
\]

where \( k \) denotes the samples composing the phoneme \( i \). The NPC structure leads to the following cost function:

\[
L = \sum_{i} \sum_{k} \left( y_{ik} - H_{a_{ij}} \circ G_w(y_{ik}) \right) \delta_{i-j}
\]

where \( H_{a_{ij}} \) is the output layer weights linked to the phoneme \( j \) and \( \delta \) the Kronecker symbol.

The weights of the output layer are proper to each phoneme, and constitute the coding coefficients, while the weights of the first layer are common to all the phonemes, and constitute the fixed part of the system.

The learning process is decomposed into two phases. The first phase is the first layer adjustment (computation of the fixed part of the coder) called the parameters adjustment phase, and the second the output weights adjustment called the coding phase.

Once the first layer weights are optimized, they remain fixed during the coding phase: the first layer acts as a simple signal transformation. Only the output layer weights are updated to minimize the error prediction.

A very simple modification of the NPC cost function allows to take into account classes membership during the parameters adjustment phase. This is done by limiting the set of the output layer weights (the coding coefficients) to one coefficient vector by phoneme class instead of one by phonemes as for NPC.

Let \( C_i \) be the class membership of phoneme \( i \) among a set of \( M \) possible classes. The cost function must be rewriting as:

\[
L = \sum_{i} \sum_{k} \left( y_{ik} - H_{a_{ij}} \circ G_w(y_{ik}) \right) \delta_{c_i-j}
\]

where

Once the prediction is minimized over all the sample sequences, the parameters (hidden layer weight) are ready to be used. The coding of any phoneme can be then performed, in the same way as those of the NPC, by

\[ y(k-L) \quad y(k-L+1) \quad y(k-L+2) \quad y(k-1) \quad y(k) \]

Non discrimnant layers Discrimant layers

phone l

Learning algorithm

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\[ y_{ik} \]

\[ \hat{y}_{ik} \]

\[ y(k-L) \quad y(k-L+1) \quad y(k-L+2) \quad y(k-1) \quad y(k) \]

Non discrimnant layers Discrimant layers

phone l

Learning algorithm

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\( y(k) \)

\( y(k+1) \)

\( y(k+2) \)

\[ y_{ik} \]

\[ \hat{y}_{ik} \]
using the first network layer as a filter. Only the second layer weights are updated to become the features vector of the phoneme.

4 Experimental conditions
To evaluate the NPC performances we tested it on a phoneme recognition task. We will describe in this part the experimental conditions.

4.1 The database
We built a four classes phoneme base using the Darp-Timit [8] database. This database is composed of speakers speaking 10 different dialects of the United States. The phonemes are chosen among the most used: /aa/, /ae/, /ey/, /ow/ (vowels). This base is constituted by 500 examples per phoneme class. To select phonemes for each class, we checked the following conditions:

- Every phoneme, according to its duration, is divided into windows of a fixed length (256 samples), each of them being a phoneme example.
- Examples are chosen randomly among all speakers so as to model a multi speaker environment.

4.2 Traditional coding methods
Among temporal coding methods, we considered the LPC coding (Linear Predictive Coding). Concerning spectral coding, we considered MFCC coding (Mel Frequency Cepstrum Coding). This coding reproduces the signal spectrum with a scale of frequencies based on the human ear scale, also called Mel frequency scale. It is among the most commonly used methods for speech processing. The encoders used provide a set of representative coefficients for each phoneme (12 coefficients in our simulations).

4.3 Classification with MLP
The classifier used to estimate performances of NPC and other coding method is a basic MLP (12 inputs, 10 hidden neurons and 4 outputs), each output corresponding to one class (4 outputs for the 4 classes). The desired output for a sample of the class 1 is therefore [+1 -1 -1 -1]. The learning rule is the gradient descent using error back propagation algorithm. After the coding phase, each window provided one sample for the classifier. The prediction memory L (the width of the prediction window) of the NPC is fixed to 20.

5 NPC evaluation using MLP classifier
In this paragraph we resume the study of the phoneme coding. Our aim is to test the possible impacts of the NPC encoder on the data classification. So we built two databases; one for the learning phase and the other for the test phase. We can then obtain a measure of the classifier generalisation capabilities.

![Fig. 3](image1.png)

One can see on figure 3 comparisons between recognition rates obtained by MLP classifier. Recognition rates have been obtained after 30000 learning iterations. The NPC coders give better results in generalisation: 62.95% for NPC-2, 61% for NPC and 58.25% for MFCC and 56.33% for LPC. Moreover, one can see the better performance of NPCII.

5.1 Evaluation of the NPC coding on b-d-g phonemes
Phonemes /b/, /d/, /g/ are of particular interest for the evaluation of coding methods in speech recognition. They frequently appear in the English language and their identification is considered to be difficult. In particular they are composed of non linear features because they belong to the set of the occlusive phonemes.
For example, those phonemes have been used by Lang and Waibel in [9],[10] to validate their model (the Time Delay Neural Network, TDNN) applied to speech signal treatment.

![Fig. 4](image2.png)

For example, those phonemes have been used by Lang and Waibel in [9],[10] to validate their model (the Time Delay Neural Network, TDNN) applied to speech signal treatment.
The NPC and the NPC-2 give as well better results for the /b/, /d/, /g/ phonemes than other methods. The non-linear features present in the speech signal are then better taking into account by the NPC coding. Moreover, the new optimisation algorithm gives better results because the classes membership was taking into account at the coding stage.

6 Discrimination ratio

There is no guarantee that the NPC-2 coder should behave as a discriminant features extractor since no mathematical proof has been done yet. Nevertheless our simulations show in an empirical way that such is the case. In addition, we defined a new discrimination measure called the discrimination ratio. Following the Itakura distance definition [11], one can define a new NPC distance between two phonemes $i$ and $j$ (or even two classes of phonemes): the ratio between the prediction error of the phoneme $i$ with the NPC model $H_{ai}$ and the prediction error of the phoneme $i$ using its own NPC model $H_{aj}$: 

$$d_{src}(i,j) = \frac{L_i}{L_j}.$$ 

We extended this measure to all the phonemes $i$ of the database by defining the ratio between the error prediction $L$ (already defined) using the NPC models $H_{ai}$ and the error prediction $Q$ of all phonemes $i$ using the NPC models $H_{aj \neq i}$:

$$Q = \sum \sum (y_{i,k} - H_{aj} \rho G_w(y_{i,k})) (1-\delta_{ci,j})$$

Thus, the discrimination ratio is given by $E = \frac{Q}{L}$.

This measure has been done during the parameter adjustment phase and during the coding phase. One can see on figure 5 and 6, the discriminant error during the coding phase for the first vowels basis and for /b/, /d/, /g/ phonemes.

We can see on those figures an increase of the discrimination ratio. It shows how during the learning time the discriminant features extraction capacity of the coder is progressively enhanced and so corroborates our first assumption.

Another interesting fact concerns the discrimination ratio measure: we can notice on figures 5 and 6 that the discrimination performances of the encoder decrease after a maximum.

This fact has been also observed in phoneme classification. During the coding phase, we stopped the iterative optimization for several iterations numbers. We computed then the recognition rate for each of these numbers with the MLP classifier. As we can observe on figure 7 and 8, the best recognition rate do not corresponds to the last coding iteration number. There is an over fitting during the coding process. The optimal iteration number is between 5 and 30. Other simulations showed the stability of this number.

Over fitting is a well known problem in neural networks. To circumvent the difficulty, we can use different strategies:

- The first is the early stopping. When the minimum error is reached on new examples, one stops the optimisation. The stop criterion we chose is the number of iterations as in the experience presented above.
• The second strategy is to optimize the weights with noisy data.
• And the last one is to add a regularization term to the error criterion. This term could be taken account the performance in classification.

![Generalisation](image)

**Fig. 8** Recognition rate for /b/, /d/, /g/ phonemes as a function of the coding iteration number

7 Conclusions and future prospects

We have presented the Neural Predictive Coding method, with the two optimisation methods NPC and NPC-2. The incorporation of database information on the NPC coder and its non-linear characteristic allow a significant improvement of the recognition rate in respect of all the other coding tested. Moreover, we show that we can make discriminative layers with a parameter adjustment phase which include the class membership notion. This optimization increased the performance in classification. We have also pointed out the over fitting problem during the coding phase, and have suggested that a solution should be to do the analysis of the discrimination ratio during the coding phase. NPC gives a new solution to the discriminant features extraction. According to these results, it seems to us a good prospective way to follow.

References: